University at Buffalo
Department of Computer Science and and Engineering
CSE 573 - Computer Vision and Image Processing

Fall 2023

Home Work #3
Due Date: 11/08/23, 11:59PM

1 Instructions

- Please submit a zip file with the name `<UBID>_hw3.zip` on UBlearns.
- `<UBID>_hw3.zip` contains one folder with the title 'Coding'.
- Follow the sub-folder structure mentioned in section 2 for the 'Coding' folder.

2 Coding

2.1 Part A: Color Conversion (40 Points)

Given a RGB image (`Lenna.png` provided in `homework_3.zip`), your program should be able to conduct the following operations and output a resultant image.

- Convert RGB to HSV: Convert the original image (`Lenna.png`) to HSV color space and save the output as `hsv_image_1.png` (10 points) (use formula mentioned in link).
- Convert RGB to HSV: Convert the original image (`Lenna.png`) to HSV color space and save the output as `hsv_image_2.png` (10 points) (use formula mentioned in class).
- Convert RGB to CMYK: Convert the original image (`Lenna.png`) to CMYK color space and save the output as `cmyk_image.png` (10 points). (use formula mentioned in class); (Use R=C, G=M, B=Y, A=K and store RGBA image as png.)
- Convert RGB to LAB: Convert the original image (`Lenna.png`) to LAB color space and save the output as `lab_image.png` (use formula mentioned in link) (10 points).

2.2 Part B: Image Filtering (40 Points)

Given a gray-scale image (`Noisy_image.png` provided in `homework_3.zip`), your program should be able to apply the filter mentioned and output a resultant image.

- Convolution: Convolve the following filter to the original image (`Noisy_image.png`) and save the output as `convolved_image.png` (7.5 points).
  $\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$
- Averaging Filter: Apply the following filter to the original image (`Noisy_image.png`) and save the output as `average_image.png` (7.5 points).
  $\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$
- Gaussian Filter: Apply the following filter to the original image (`Noisy_image.png`) and save the output as `gaussian_image.png` (7.5 points).
  $\frac{1}{16}\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix}$
- Median Filter: Apply a 5×5 median filter to the original image (`Noisy_image.png`) and save the output as `median_image.png` (7.5 points).
- Contrast and Brightness Adjustment: Apply contrast and brightness adjustment so that the original image (`Uexposed.png` in `homework_3.zip`) changes to high contrast and brighter image (changes should be visibly perceivable) and save the output as `adjusted_image.png` (10 points).

2.3 Part C: Gaussian Filter Smoothing using Fourier transform (20 Points)

Given a gray-scale image (`Noisy_image.png` provided in `homework_3.zip`), your program should be able to do the following steps.

- Convert gray-scale image to Fourier domain: Convert the original image (`Noisy_image.png`) to Fourier domain and save the output as `converted_fourier.png` (10 points).
- Gaussian Filter smoothing: First, apply a low-pass filter (you can choose the filter that achieves reasonable results) in the Fourier domain. Then, invert the image to gray-scale space and save the output as `gaussian_fourier.png` (10 points).
  Note: The results of `guassian_fourier.png` and `gaussian_image.png` (from part B) should look very similar in terms of outputs.

2.4 OpenCV Libraries permitted and prohibited

- You are allowed to use OpenCV version $\geq$ 4.5.4 for this homework.
- You may only use OpenCV APIs for reading (`cvtColor(COLOR_BGR2RGB)` to read) and writing an image. All other manipulations of the image have to be manually coded in python.
- Specifically, you may not use **ANY openCV functions** that can directly accomplish the task. For example, you may not use `cv2.cvtColor(image, cv2.COLOR_RGB2HSV)`, `cv2.cvtColor(image, cv2.COLOR_CMYK2LAB)`, `cv2.cvtColor(image, cv2.COLOR_HSV2CMYK)`, `cv2.cvtColor(image, cv2.COLOR_LAB2RGB)`, `cv2.filter2D`, `cv2.blur`, `cv2.boxfilter`, `cv2.GuassianFilter`, `cv2.medianBlu` `cv2.convertScaleAbs`, `cv2.adjustGamma`.
- You may use `cv2.addWeighted`.
- **ONLY** for fourier transform, you may use `cv2.dft`, `cv2.idft`, `cv2.magnitude` and **ANY** function in numpy.

2.5 Submission Folder Structure

Please submit the code written in python as `color_conversion.py`, `image_filtering.py` and `fourier_filtering.py` for part A, part B and part C respectively.

- `UBID_hw3.zip`
  - `Coding`
    - `Lenna.png`
    - `hsv_image_1.png`
    - `hsv_image_2.png`
    - `cmyk_image.png`
    - `lab_image.png`
    - `Noise_image.png`
    - `convolved_image.png`
    - `average_image.png`
    - `gaussian_image.png`
    - `median_image.png`
    - `Uexposed.png`
    - `adjusted_image.png`
    - `converted_fourier.png`
    - `guassian_fourier.png`

3 Submission Guidelines

- Unlimited number of submissions is allowed and only the latest submission will be used for grading. Create `<UBID>_hw3.zip` and upload it to UBlearns.
- Identical code will be treated as plagiarism. Please work it out independently. Using online code or manipulated images available online will be considered as plagiarism.
- For code raising "RuntimeError", the grade will be ZERO for the homework.
- Any variations to the above submission folder structure will result in a ZERO for the homework. Also, there is no need to use any hard-coded local paths in your homework. Usage of such paths, would break when we run your code and will result in a ZERO.
- Late submissions guidelines apply for this homework.